

# COMPONENT RECORDING METHOD FOR COMPUTER PROGRAM AND DEVICE THEREFOR

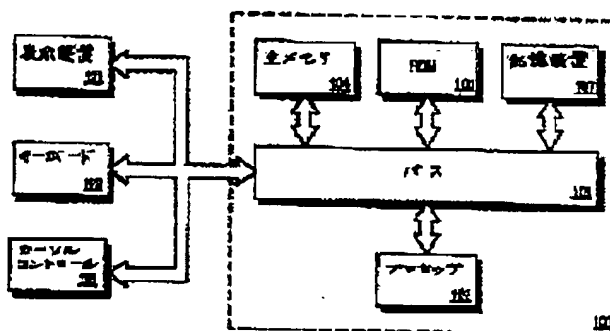
Patent number: JP9223024  
 Publication date: 1997-08-26  
 Inventor: BAATO SUMAARUDAASU; KEBIN JIEI KURAAKU  
 Applicant: SUN MICROSYSTEMS INC  
 Classification:  
 - international: G06F9/45; G06F9/06  
 - european: G06F9/445; G06F9/45E3T; G06F9/45M1L; G06F11/34T  
 Application number: JP19960207692 19960719  
 Priority number(s): US19950504091 19950719

Also published as:  
 EP0755003 (A)

Report a data error here

## Abstract of JP9223024

**PROBLEM TO BE SOLVED:** To carry out the same operations with less capacity of a high speed memory by detecting one of plural code sections that is used in an execution period and sequencing these code sections based on the used one. **SOLUTION:** In the execution of a computer program, a computer system 100 detects specific one of code sections of an instruction data image that is accessed by a processor 102 and also detects the time when the code section is accessed. These access records are stored in an optional storage such as a disk 107 or a memory 104. When the system 100 completes its all writing operations to an access matrix based on the access data stored in an execution period, a binary array corresponding to a specific code section is stored in each row. Then, the order of the code sections is determined based these binary arrays.



Data supplied from the esp@cenet database - Worldwide

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-223024

(43) 公開日 平成9年(1997)8月26日

(51) Int. Cl. <sup>6</sup>	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 9/45			G 0 6 F 9/44	3 2 2 F
9/06	4 1 0		9/06	4 1 0 E

審査請求 未請求 請求項の数 7 F D (全 13 頁)

(21) 出願番号 特願平8-207692

(22) 出願日 平成8年(1996)7月19日

(31) 優先権主張番号 08/504091

(32) 優先日 1995年7月19日

(33) 優先権主張国 米国 (US)

(71) 出願人 591064003

サン・マイクロシステムズ・インコーポレ  
ーテッド

SUN MICROSYSTEMS, IN  
CORPORATED

アメリカ合衆国 94043 カリフォルニア  
州・マウンテンビュー・ガルシア アヴェ  
ニュー・2550

(72) 発明者 パート・スモールダース

アメリカ合衆国 95125 カリフォルニア  
州・サン ホゼ・ジョナサン アヴェニ  
ュ・1737

(74) 代理人 弁理士 山川 政樹

最終頁に続く

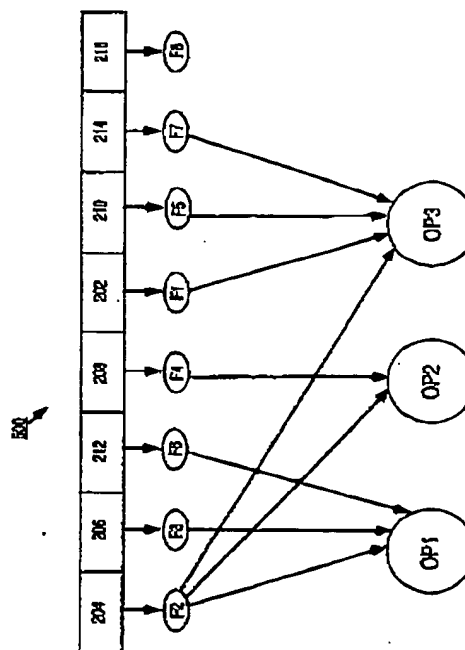
(54) 【発明の名称】 コンピュータプログラムのコンポーネントを記録するための方法及び装置

(57) 【要約】

【課題】 コンピュータプログラムのコードセクションを順序付けるための方法及び装置を提供する。

【解決手段】 実行期間中にコンピュータプログラムを実行し、その実行期間中にコンピュータプログラムの複数の各コードセクションのうちどのコードセクションが使用されたかを検知し、実行期間中に各コードセクションのうちどのコードセクションが使用されたかに基づいてコードセクションを順序付ける。

【効果】 本発明によれば、同じ動作をより少ない高速メモリ容量を用いて実行することが可能になる。



## 【特許請求の範囲】

【請求項1】 複数のコードセクションを有するコンピュータプログラムのそれらのコードセクションを順序付けるためのコンピュータシステムにおいて実行可能な方法において、

実行期間中にそのコンピュータプログラムを実行するステップと、

その実行期間中に複数の各コードセクションのうちどのコードセクションが使用されたかを検知するステップと、

その実行期間中に各コードセクションのうちどのコードセクションが使用されたかに基づいてコードセクションを順序付けるステップと、を具備した方法。

【請求項2】 前記実行期間を複数の時間間隔に分割するステップと、

前記各コードセクションが使用された複数の時間間隔に基づいてコードセクションを順序付けるステップと、をさらに具備した請求項1記載の方法。

【請求項3】 コンピュータプログラムの複数のコードセクションを記憶するメモリと、

そのメモリに接続されたプロセッサであって、実行期間中に複数のコードセクションにアクセスしてその実行期間中にコンピュータプログラムを実行し、その実行期間中に複数のコードセクションの中特定のコードセクションがあるプロセッサによってアクセスされたかを指示するアクセス記録を前記メモリに記憶するプロセッサと、

そのアクセス記録に基づいて複数のコードセクションを順序付けるよう構成されたメカニズムと、を具備したコンピュータシステム。

【請求項4】 複数のコードセクションを有するターゲットコンピュータプログラムのそれらのコードセクションを順序付けるためのコンピュータ可読コードが実装されたコンピュータ使用可能媒体と、

ターゲットコンピュータプログラムが実行される実行期間中に複数の各コードセクションが使用された時これをコンピュータに検知させるよう構成されたコンピュータ可読プログラムコードデバイスと、

その実行期間中に各コードセクションが何時使用されたかに基づいてコンピュータにコードセクションを順番付けさせるよう構成されたコンピュータ可読プログラムコードデバイスと、を具備したコンピュータプログラム製品。

【請求項5】 コンピュータに前記実行期間を複数の時間間隔に分割させるよう構成されたコンピュータ可読プログラムコードデバイスと、

その複数の時間間隔のうち各コードセクションが使用された時間間隔に基づいてコンピュータにコードセクションを順番付けさせるよう構成されたコンピュータ可読プログラムコードデバイスと、をさらに具備した請求項4

記載のコンピュータプログラム製品。

【請求項6】 コンピュータプログラムによりアクセスされる複数のデータセクションを有するデータを順序付けるコンピュータシステム上で実行可能な方法において、

実行期間中にコンピュータプログラムを実行するステップと、

その実行期間中に複数の各データセクションがコンピュータプログラムによりアクセスされた時これを検知するステップと、

前記実行期間中に各データセクションが何時アクセスされたかに基づいてデータセクションを順序付けるステップと、を具備した方法。

【請求項7】 前記実行期間を複数の時間間隔に分割するステップと、

その複数の時間間隔のうち各データセクションがコンピュータプログラムによってアクセスされた時間間隔に基づいてデータセクションを順序付けるステップと、をさらに具備した請求項6記載の方法。

## 【発明の詳細な説明】

【0001】

【発明が属する技術分野】本発明は、ソフトウェアの開発技術に関し、より詳しくは、コンピュータプログラムのコードセクションを仮想メモリ環境における効率的な実行が達成されるよう順序付けるための方法及び装置に関する。

【0002】

【従来の技術】コンピュータプログラムには、アプリケーションプログラム、デバイスドライバ、ライブラリ等、数多くの種類がある。説明の便宜上、以下本願においては、このような全ての種類のコンピュータプログラムをまとめて、「コンピュータプログラム」と総称する。プログラムの種類に関わらず、あらゆるコンピュータプログラムは、プロセッサによって実行されたとき、プロセッサに何らかの関数を遂行させる命令を含む。コンピュータプログラムが実行されていないときは、命令を表すデジタルデータ（「命令データイメージ」）は通常磁気ディスクまたは光ディスクのような比較的低速の不揮発性記憶装置に記憶されている。記憶装置が低速であることは、コンピュータプログラムの実行時に命令が記憶装置から直接読み出される場合、相当大きなネックになる。

【0003】このようなネックを回避するために、通常、命令データイメージはプログラムの実行に先立ってランダムアクセスメモリのような比較的高速の揮発性記憶装置にコピーされる。プロセッサは、そのような比較的高速の記憶装置に入れられた命令データイメージのコピーから直接命令を読み込み、これによって低速の記憶装置に常時アクセスする必要が回避される。

【0004】コンピュータが遂行する関数が複雑になる

にしたがって、命令データイメージのサイズも大きくなる。その結果、プログラム実行時に命令データイメージのコピー全体を記憶するために必要な高速メモリの量も増大する。高速メモリは、比較的高価であるため、プログラムを効率的に実行するのに必要な高速メモリの量を少なくするための技術が開発されてきた。

【0005】一つの技術によれば、高速記憶装置の利用可能な量よりも大きい「仮想メモリ」を得るために、近い方の記憶装置の一部を遠い方の記憶装置と結合することが行われている。プログラム実行時には、そのプログラムの命令データイメージは仮想メモリに記憶される。通常、このことは、任意の所与の時点において、命令データイメージのある部分は低速記憶装置に記憶され、命令データイメージの他の部分は高速記憶装置に記憶されるということの意味する。本願においては、高速記憶装置に記憶される命令データイメージの部分をコンピュータプログラムの「ロードされた部分」と称する。高速記憶装置の無駄使いを避けるためには、コンピュータプログラムのロードされた部分を現在の高レベル・オペレーションを実行するのに必要な命令データイメージの部分のみに限定する方が好ましい。

【0006】典型的には、命令は高速記憶装置にロードしてからでないと実行することができない。そのため、命令を実行する前には、その命令を表す命令データイメージの部分はロードされた部分の一部でなければならない。もしその命令を表す命令データイメージがまだ高速記憶装置に記憶されていないならば、低速記憶装置から高速記憶装置へコピーされる。その時、もし高速記憶装置に十分な未使用空間がなければ、高速記憶装置の一部に必要な命令データイメージを受け入れられるように解放しなければならない。高速記憶装置の一部を解放するプロセスは、高速記憶装置から低速記憶装置への変更されたデータのコピー動作を伴う場合がある。低速記憶装置と高速記憶装置との間でデータが転送される都度、コンピュータプログラムの実行が遅延する。

【0007】仮想メモリの低速部と高速部との間で行われるデータ転送の回数をできるだけ少なくするための技術が開発されている。具体的に言うと、プロセッサに低レベルオペレーションを行わせる命令は、通常、命令データイメージ中で互いに近くに置かれるということがわかった。従って、一つの命令が実行される時は、近い将来その命令の近くにある他の命令が実行されることが多い。その結果、ほとんどの仮想メモリシステムは、仮想メモリの低速部と高速部との間で、命令単位ではなく、ブロック単位で命令を移動させるようになっている。

【0008】高レベルオペレーションに対応する命令を強制的に同じコードブロックにまとめることによって、仮想メモリの低速部と高速部との間のデータ転送回数も、ロードされた部分のサイズも小さくすることができる。例えば、関数X、Y及びZの実行が必要な独立の高

レベルオペレーションOを考えて見よう。もし、関数X、Y及びZを実行するための命令が同じコードブロックにあれば、最小のロードされた部分は単一ブロックからなると思われる。さらに、オペレーションOの実行前に行われるコードブロック転送は最大1回であろう。しかしながら、関数X、Y及びZを実行するための命令シーケンスが各々別個のコードブロックに記憶されている場合は、オペレーションOの実行時の最小のロードされた部分には3ブロックが含まれることになる。さらに、ロードされた部分をロードするのに最大3回のコードブロック転送が必要な場合がある。また、関数Xのための命令が多くコードブロックに分散し、関数Yのための命令が多くコードブロックに分散し、関数Zのための命令が多くコードブロックに分散していれば、さらに多くのコードブロックが必要になる場合がある。

【0009】

【発明が解決しようとする課題】上記の事実に基づいて考えると、最小のロードされた部分のサイズを小さくするための方法及び装置を提供することが望ましい。さらに、仮想メモリ環境におけるコンピュータプログラムの実行時に行われるコードブロック転送の回数を少なくするための方法を提供することが望ましい。さらに、コンピュータプログラムの命令データイメージのセクションを独立の高レベルオペレーションと比較的高い相関を有するグループにまとめるための方法及び装置を提供することが望ましい。

【0010】

【課題を解決するための手段】本発明は、コンピュータプログラムのコードセクションを順序付けるための方法及び装置である。コンピュータプログラムは試行的実行期間中に実行される。実行期間中にアクセス記録が作成される。そのアクセス記録は、各コードセクションが何時アクセスされたかを指示する。実行期間終了後、コードセクションは、それらが実行期間中にあるアクセスされたかに基づいて記録される。

【0011】本発明の一態様によれば、実行期間は複数の時間間隔に分割される。各時間間隔がコンピュータプログラムの独立の高レベルオペレーションが実行された期間に対応する。そして、各行がコードセクションに対応し、各列が時間間隔に対応するアクセスマトリクスが生成される。対応する時間間隔中に対応するコードセクションがアクセスされたかどうかに基づいて、データマトリクスの各セルに値が記憶される。次に、コードセクションが、対応する行の値に基づいて記録される。

【0012】本発明のもう一つの態様によれば、アクセスマトリクスの各セルは1つのビットを記憶する。その結果、各コードセクションは対応するビットアレイを有する。コードセクションは、それらの対応ビットアレイによって表される数の数値の順序に基づいて記録することができる。あるいは、コードセクションは、連続する

コードセクションのビットアレイ間の偏差の度合が最小になるようにして記録することも可能である。

【0013】以下、本発明を添付図面に示す実施形態により説明するが、これらの実施形態は本発明を限定するためのものではない。図中、同じ符号は類似の要素を指す。

【0014】

【発明の実施の形態】以下、コンピュータプログラムのコードセクションを順序付けるための方法及び装置の実施形態について説明する。以下の説明においては、本発明の完全な理解を図るために、説明の便宜上特定の詳細事項が多数記載されている。しかしながら、本発明がこれら特定の詳細事項の記載なしで実施可能であることは明白であろう。その他の場合においては、周知の構成及び装置は、本発明を不必要にあいまいにすることがないように、ブロック図形式で示されている。

【0015】図1においては、本発明の実施形態を実装することができるコンピュータシステムが100で示されている。コンピュータシステム100は、情報を伝送するためのバスまたはその他の伝送手段101、及び情報を処理するためのバス101に接続された処理手段102を有する。コンピュータシステム100は、さらに、バス101に接続されていて、プロセッサ102により実行される情報及び命令を記憶するためのランダムアクセスメモリ(RAM)またはその他のダイナミック記憶装置104(主メモリと称する)を有する。主メモリ104は、プロセッサ102による命令実行時に一時的数値変数及び命令を記憶するために使用することもできる。また、コンピュータシステム100は、バス101に接続されていて、プロセッサ102用の静的情報及び命令を記憶するためのリードオンリーメモリ(ROM)及び/またはその他の静的記憶装置106を有する。バス101には、情報及び命令を記憶するためのデータ記憶装置107が接続されている。

【0016】磁気ディスクまたは光ディスクのようなデータ記憶装置107及びこれに対応するディスクドライブはコンピュータシステム100に接続することができる。また、コンピュータシステムは、バス101を介して、コンピュータユーザに情報を表示するためのブラウン管(CRT)のようなディスプレイ装置121に接続することもできる。通常は、プロセッサ102に情報を伝達し、コマンド選択を行うために、アルファニューメリックキー及びその他のキーを含むアルファニューメリック入力装置122がバス101に接続される。それ以外の種類のユーザ入力装置として、プロセッサ102に対して方向指示情報を入力し、コマンド選択を行うとともに、ディスプレイ121上のカーソルの動きを制御するためのマウス、トラックボールあるいはカーソル方向キーのようなカーソルコントロール123が接続されている。この入力装置は、通常、平面内のいちを指定する

ことができるよう、2軸、すなわち第1の軸(例えばx軸)と第2の軸(例えばy軸)における2自由度を有する。

【0017】あるいは、スタイラスまたはペンのような他の入力装置を用いてディスプレイと対話することもできる。コンピュータスクリーン上に表示されたオブジェクトは、それらの表示されたオブジェクトにタッチするためのスタイラスまたはペンを用いて選択することができる。コンピュータは、その選択されたオブジェクトをタッチ感知スクリーンを用いて検出する。同様に、ライトペンや感光スクリーンを用いて表示されたオブジェクトを選択することもできる。この種の入力装置では、マウスやトラックボールを用いたシステムにおけるように「ポイント・アンド・クリック」操作と異なり、選択位置と選択されたオブジェクトを単一操作として検出することができる。スタイラス及びペン型入力装置やタッチスクリーン及び感光スクリーンは、当技術分野においては周知である。このようなシステムにおいては、インタフェースがすべて筆記具(ペンのような)としてのスタイラスを介して与えられ、書かれたテキストが光学文字認識装置(OCR)を用いて変換され、122で示するようなキーボードがない場合もある。

【0018】本発明の一実施形態によれば、コンピュータシステム100はコンピュータプログラムのコードセクションの順序を決定するよう構成される。この順序は高レベルオペレーションを実行させるコードセクションを一つにまとめさせるものである。その結果、最小にロードされた部分におけるブロック数が少なくなる。さらに、仮想メモリを有するシステムでコンピュータプログラムが実行させるとき起こるコードブロック転送の回数も少なくなる。

【0019】図2には、一例のコンピュータプログラムPの命令データイメージ200が示されている。コンピュータプログラムPが実行されていないときは、命令データイメージ200は通常ディスク107のような低速記憶装置に記憶される。コンピュータプログラムPが実行されるときは、命令データイメージ200は、その一部または全部がメモリ104のような高速記憶装置にロードされる。

【0020】命令データイメージ200は、コードセクション202、204、208、210、212、214及び216を含む。説明の便宜上、コードセクション202、204、208、210、212及び216はそれぞれ関数F1、F2、F3、F4、F5、F6、F7及びF8を実装すると仮定する。また、コンピュータプログラムPは、独立の3つの高レベルオペレーション、すなわちOP1、OP2及びOP3を実行するものと仮定する。本願において、「独立のオペレーション」という用語は、いったん呼び出されると、割り込みがかけられない限り完遂されるオペレーションを指すものとす

る。このようなオペレーションとしては、例えば、プログラム初期化オペレーション、演算オペレーション及びプログラム終了オペレーション等がある。また、オペレーションOP1は、関数F2、F3及びF6の実行を必要とし、オペレーションOP2は、関数F2及びF4の実行を必要とし、オペレーションOP3は、関数F1、F2、F5及びF7の実行を必要とするものと仮定する。この場合、関数F8はコンピュータプログラムPによって実行されるどのオペレーションでも必要とされない。従って、関数F8を実行するための命令を含むコードセクション216は全くアクセスされない。

#### 【0021】モニタ下におけるプログラムの実行

本発明の一実施形態によれば、コンピュータプログラムPはコンピュータシステム100上で実行される。具体的に言うと、ユーザはコンピュータシステム100にコンピュータプログラムPの実行を開始させるための入力を入れる。すると、前記の仮定により、ユーザ入力にตอบสนองしてプログラム初期化オペレーションOP1が実行される。

【0022】図3には、プロセッサ102がコンピュータプログラムPを実行している期間中に起こる事象のタイムライン300が示されている。時刻T0は、ユーザが入力を入れてコンピュータプログラムPを呼び出す時刻を表している。プロセッサ102は、ユーザ入力にตอบสนองしてオペレーションOP1を実行する。OP1の実行中、プロセッサ102は、それぞれ線302、304及び306によって示すように、コードセクション206、204及び212に入っている命令を読み出し、実行することによって関数F3、F2及びF6を実行する。

【0023】図3では、206、204及び212の各コードセクションへのアクセスを1本の線で表してある。しかしながら、1つの関数の実行には、ある特定コードセクションに対するアクセスが何100回も必要な場合があるということに留意すべきである。一般に、あるコードセクションの最初のアクセスは、そのコードセクションを含むコードブロックを高速記憶装置にロードさせる。同じ独立の高レベルオペレーション中におけるその同じコードへの以後のアクセスは、そのコードセクションを含むコードブロックがその高レベルオペレーションが行われている間高速メモリ外に転送されることがないような高速でもって行われる。そのために、独立の高レベルオペレーションの実行中に特定のコードセクションがアクセスされる回数は、ロードされた部分の最小サイズ及びその高レベルオペレーションの実行中に行わなければならないブロック転送の回数にほとんど影響を及ぼさない。

【0024】コンピュータプログラムPの実行中、コンピュータシステム100は、命令データイメージ200のどのコードセクションがプロセッサ102によってア

クセスされたか、及びそれらのコードセクションがアクセスされた時刻を検出する。このデータを以下「アクセス」と称する。コンピュータシステム100は、アクセスレコードをディスク107あるいはメモリ104のような任意の記憶装置に記憶することができる。あるいは、コンピュータシステム100は、ユーザによって何時入力が入れられたかを検出し、記録するようにしてもよい。

【0025】初期化オペレーションOP1は、プロセッサ102が関数F6の実行を完了すると終了する。オペレーションOP1が終了すると、コンピュータシステム100は、ユーザが次にどの動作を実行すべきかを指示するまでコンピュータプログラムPの実行を停止する。コンピュータプログラムPの実行が停止されている間、プロセッサ102は他のコンピュータプログラムに関連する他のオペレーションを実行したり、入力装置をポーリングしてユーザ入力を検出したり、あるいはこれら両方を行うことが可能である。

【0026】説明の便宜上、時刻T1にユーザはオペレーションOP2を呼び出すための入力を入れるものと仮定する。コンピュータシステム100はこのユーザ入力に関連した時刻を検出し、記録する。プロセッサ102は、検出されたユーザ入力にตอบสนองしてオペレーションOP2を実行する。オペレーションOP2の実行中、プロセッサ102はコードセクション208に入っている命令を読み出し、実行して、関数4を実行し、またコードセクション204に入っている関数F2を実行するための命令を読み出して実行する。コンピュータシステム100は、これらの各コードセクションが何時アクセスされたかを検出し、アクセスレコードに記録する。オペレーションOP2は、プロセッサ102が関数F2の実行を完了すると終了する。オペレーションOP2が終了すると、コンピュータシステムは次にどのオペレーションを実行すべきかを決定するユーザ入力に供給されるまで待機する。

【0027】時刻T2では、ユーザはコンピュータプログラムPを終了するための入力を入れる。プロセッサ102は、このユーザ入力にตอบสนองして、コードセクション204から命令を読み出し、実行して関数F2を実行し、コードセクション210から命令を読み出し、実行して関数F5を実行し、コードセクション202からコードセクションを読み出し、実行して関数F1を実行し、コードセクション214から命令を読み出し、実行して関数F7を実行する。コンピュータシステム100は、これらの各コードセクションのアクセスタイミングをアクセスレコードに記録する。時刻T3では、命令データイメージ200に対応するコンピュータプログラムPの実行が終了する。

#### 【0028】アクセス検出

上記のモニタ下におけるプログラム実行プロセス時に

は、コンピュータシステム100は、各特定のコードセクションがアクセスされたとき、これを検出して記録する。このアクセス検出を行うためには多くの技術的方法を用いることができる。本発明の一実施形態によれば、各コードセクションには命令シーケンスが入れられる。命令シーケンスは、プロセッサ102にその命令シーケンスがあるコードセクションを識別指示するデータ及び各時点における現在時刻を記録させる。例えば、コードセクション202には、実行されると、プロセッサ102に関数F1を識別指示するデータ及び現在時刻を記録させる命令シーケンスを埋め込むことができる。同様に、コードセクション204には、実行されると、プロセッサ102に関数F2を識別指示するデータ及び現在時刻を記録させる命令シーケンスを埋め込むことができる。

【0029】本発明のもう一つの実施形態においては、コンピュータシステム100は、コンピュータプログラムPの実行と同時に検出プログラムを実行するよう構成される。検出プログラムは、コンピュータシステム100にプロセッサがコンピュータプログラムPのいろいろなコードセクションにアクセスしているとき、それを検出、記録させる。本発明のさらに他の実施形態においては、コンピュータプログラムPの実行時にプロセッサ102によってなされたコールを検出、記録するためのモニタ用ハードウェアがコンピュータシステム100に付加される。

#### 【0030】アクセスマトリクスの構築

コンピュータプログラムPの実行が終了すると、コンピュータシステム100は、コンピュータプログラムPの実行中に記憶された情報に基づいてアクセスマトリクスをメモリ104内に構築する。

【0031】最初に、コンピュータシステム100はT0とT3の間の期間（「実行期間」）をいくつかの時間間隔に分割する。実行期間を分割するこれらの時間間隔の大きさ、数、及び境界を決定するには種々の方法を用いることができる。例えば、実行期間は所定の持続時間を有する時間間隔に分割することができる。又は、実行期間は、同じ持続時間の所定数の時間間隔に分割することもできる。また、実行期間は、実行期間中の関数呼出しの密度と分布及び／またはユーザ入力タイミングのような要素に基づく可変持続時間の時間間隔に分割することもできる。本発明は、実行期間を幾つかの時間間隔に分割する各特定の技術的方法に限定されるものではない。しかしながら、これにより選択される時間間隔は、各時間間隔内で低い平均ページング速度が確保されるよう十分短くなることが望ましい。

【0032】この例においては、コンピュータシステム100は、実行期間を実行期間中のユーザ入力タイミングに基づいて時間間隔に分割する。具体的に言うと、実行期間は3つの時間間隔INT1、INT2及びINT3に分割される。

時間間隔INT1は、プログラムPを呼び出した入力とオペレーションOP2を呼び出した入力との間の時間に広がる。時間間隔INT2は、オペレーションOP2を呼び出した入力とオペレーションOP3を呼び出した入力との間の時間に広がる。時間間隔INT3は、オペレーションOP3を呼び出した入力とコンピュータプログラムPの終了との間の時間に広がる。

【0033】これらの種々の時間間隔が決定されたならば、コンピュータシステム100は、各行が命令データイメージ200中のコードセクションに対応し、各列が1つの時間間隔に対応するマトリクスを構築する。マトリクスは、例えば、メモリ104中に構築することができる。図4には、それぞれ時間間隔INT1、INT2及びINT3に対応する3本の列と、それぞれコードセクション202、204、206、208、210、212、214及び216に対応する8本の行408、410、412、414、416、418、420及び422を有するアクセスマトリクスが示されている。

【0034】コンピュータシステム100は、実行期間中に生成されたアクセスレコードに基づいてマトリクス400に値を書き込む。一実施形態によれば、アクセスマトリクス400の各セルには1ビットが書き込まれる。各セルのビットは、各セルが存在する行に対応するコードセクションが、そのセルが存在する列に対応する時間間隔中にアクセスされた場合にのみセットされる。例えば、セル424はコードセクション202に対応する行408、及び時間間隔INT1に対応する列402にある。その結果、セル424に記憶されたビットは、コード202が時間間隔INT1中にアクセスされた場合にのみセットされる。上記の例においては、コードセクション202は時間間隔INT1中にはアクセスされなかったもので、セル424には「0」が書き込まれる。

【0035】コンピュータシステムが実行期間中に記憶されたアクセスデータに基づいてアクセスマトリクス400に全て書き込み終わると、各行には特定のコードセクションに対応する2進アレイが記憶された状態になる。例えば、コードセクション202に対応する行408には、2進アレイ「001」が記憶される。次に、コンピュータシステム100は、各コードセクションに対応する2進アレイに基づいてそれらのコードセクションの順序を決定する。

#### 【0036】順序決定

各コードセクションに対応する2進アレイに基づいてそれらのコードセクションの順序を決定するには、いろいろな技術的方法を用いることができる。例えば、各コードセクションに対応する2進アレイによって表される2進数の値に基づいて各コードセクションにある数を割り当てることができる。この順序付け方法をここで説明し

ている例に適用すると、コードセクション202には、その2進アレイは“001”であるから、数“1”が割り当てられる。コードセクション204には、このコードセクション204に対応する2進アレイ“111”によって表される数“7”が割り当てられる。また、コードセクション206、208、210、213、214及び216には、それぞれ数4、2、1、4、1及び0が割り当てられる。

【0037】各コードセクションに数が割り当てられたならば、コンピュータシステム100は各コードセクションに割り当てられた数の大ききの順に、コードセクションの順序を決定する。例えば、数“7”が割り当てられたコードセクション204の次に数“4”が割り当てられたコードセクション206及び212が来る。コードセクション206及び212の後には、数“2”が割り当てられたコードセクション208が来、その後に、数“1”が割り当てられたコードセクション202、210及び214が来る。数“0”が割り当てられたコードセクション216はこの順番の最後になる。

【0038】ここで説明する例においては、複数のコードセクションが同じ2進アレイを有する。例えば、コードセクション202、210及び214は全て同じ2進アレイ“001”を有する。そのために、これらのコードセクションを互いにどのように順序付けるかを決定するには2進アレイ以外の何かを利用しなければならない。一実施形態によれば、同じ2進アレイを有するコードセクション間の順序はランダムに行われるだけである。もう一つの実施形態においては、同じ2進アレイを有するコードセクションは、それらのコードセクションが1つまたは2つ以上の時間間隔中にアクセスされた順序に基づいて順序付けられる。例えば、コードセクション202、210及び214は全て時間間隔INT3中にアクセスされている。これらの3つのコードセクションの中では、コードセクション210が最初にアクセスされ、コードセクション202が次にアクセスされ、コードセクション214が3番目にアクセスされる。従ってこのアクセス順をコードセクションの順序付けに反映することができる。

【0039】図5は、上記のプロセスによって決定された順序に従い命令データイメージ200のコードセクションを順序付けることにより構築された命令データイメージ500を示す。ここで、命令データイメージ500のコードセクションの位置とそれらのコードセクションを使用する動作との間には、命令データイメージのコードセクションの位置とそれらのコードセクションを使用する動作との間よりも強い相関があるということに留意すべきである。このようなより強い相関の結果、命令データイメージによって表されるコンピュータプログラムPが仮想メモリシステムで実行される場合、相当大きな性能の改善が達成される。

【0040】例えば、202、204、206、208、210、212、214及び216の各コードセクションのサイズは1Kバイトであると仮定する。また、コンピュータプログラムPは、仮想メモリの低速部と高速部との間で4Kブロック単位で転送を行う仮想メモリシステム上で実行されるものと仮定する。これらの条件下では、8つのコードセクションは2つの4Kブロックにまたがって分布する。コードセクション202、204、206、208、210、212、214及び216が命令データイメージ200に示されているように順序付けられるとすると、第1ブロック（ブロック1）には、コードセクション202、204、206及び208が入り、第2ブロック（ブロック2）には、コードセクション210、212、214及び216が入ることになる。コードセクション202、204、206、208、210、212、214及び216が命令データイメージ500に示されているように順序付けられるとすると、第1ブロック（ブロックA）には、コードセクション204、206、212及び208が入り、第2ブロック（ブロックB）には、コードセクション202、210、214及び216が入ることになる。

【0041】上記に基づいて、コードセクションが命令データイメージ200に示されているように順序付けられとすると、オペレーションOP1の実行には2つの4Kブロック（ブロック1及びブロック2）のローディングが必要であり、オペレーションOP2の実行には1つの4Kブロック（ブロック1）のローディングが必要であり、オペレーションOP3の実行には2つの4Kブロック（ブロック1及び2）のローディングが必要である。これに対して、コードセクションが命令データイメージ500に示されているように順序付けられるとすると、オペレーションOP1を実行するには1つのブロック（ブロックA）だけロードすればよく、オペレーションOP2を実行するには1つのブロック（ブロックA）をロードするだけでよく、オペレーションOP3を実行するには2つのブロック（ブロックA及びブロックB）をロードするだけでよい。このようにして、コードセクションが仮想メモリの高速記憶部に入っていない場合、仮想メモリの低速記憶部から高速記憶部にコピーすることによってコードセクションが「ロードされる」。

【0042】コードセクションが命令データイメージ500に示されているように順序付けられると、初期化オペレーションOP1実行時にロードするブロックが1つ少なく済むことによって、相当大きい効率改善が達成される場合がある。例えば、2ブロック初期化オペレーションOP1の場合に必要な可能性のある余分のオペレーションについて考えてみよう。仮想メモリの高速部がすでにいっぱい書き込まれているとすると、ブロック2を書き込むためのスペースを得るために1ブロックのデータを仮想メモリの低速記憶部に書き込まなければ



ばならない場合がある。その場合、OP 1が終了すると、ブロック2がまさしく高速記憶部の貴重なスペースを、割り当て解除されるまで、あるいはコンピュータプログラムPが終了するまで占有することになる。

【0043】ここで、簡単な例を取り上げて、本発明をわかりやすく説明した。しかしながら、多くのコンピュータプログラムはここで説明するコンピュータプログラムPの例よりかなり複雑である。実際、多くのコンピュータプログラムは、数百の独立した高レベルオペレーション及び数千の関数を実行する。このような複雑な条件においては、本発明によりコードセクションを順序付けることによってより大きな効率改善を達成することができる。例えば、複雑な動作を行うコードセクションは、それらのコードセクションを順序付ける前に10の4Kデータブロックにわたって分布する場合がある。そして、本発明の順序付けによって、コードセクションを1つの4Kデータブロックにまとめることができる。このように、上記のオペレーションを実行するには、高速記憶部に10分の1の数のデータブロックをローディングしさえすればよい。

【0044】ビットパターンの類似性に基づく順序決定上に説明した順序付け方法は、アクセスレコードに基づいてコードセクションを順序付けるのに可能な数多くの方法の中の1つでしかない。本発明のもう一つの実施形態によれば、アクセスマトリクス400は上に説明したようにして構築されるが、コードセクションは、連続するコードセクションのビットアレイ間でできるだけ少ない数のビットが異なるようにして順序付けられる。

【0045】このような順序に達するには、最初のコードセクションを選択し、その新しいコードセクション順序の最初に置く。この第1のコードセクションに対応するビットアレイが「現在のビットアレイ」になる。この現在のビットアレイを他のコードセクションのビットアレイと比較して、現在のビットアレイとの差違が最も小さいビットアレイを決定する。この場合、2つのビットアレイで、対応するビット値が何ビット異なるかによってそれら2つのビットアレイ間の差違度が決定される。例えば、コードセクション202のビットアレイ“001”中の3ビットはコードセクション210のビットアレイ“001”の3ビットと同じである。その結果、これら2つのビットアレイ間の差違度はゼロである。これに対して、コードセクション202のビットアレイ“001”は、第2ビットだけがコードセクション206のビットアレイ“100”の対応ビットと一致する。従って、コードセクション202と206のビットアレイ間の差違度は2である。差違度は、2つのビットアレイの排他的ORでセットされるビット数をカウントすることにより得ることができる。

【0046】現在のビットアレイとの差違度が最も小さいビットアレイが決まると、それが現在のビットアレイ

になり、そのビットアレイに対応するコードセクションが新しいコードセクション順序の2番目の位置に置かれる。次に、その新しい現在のビットアレイが新しいコードセクション順序にまだ入れられていない全てのコードセクションのビットアレイと比較される。そして現在のビットアレイとの差違度が最も小さいビットアレイが新しい現在のビットアレイになり、そのビットアレイに対応するコードセクションが新しいコードセクション順序の3番目の位置に置かれる。このプロセスが、全てのコードセクションが新しいコードセクション順序に並べられるまで繰り返される。

【0047】この順序付け方法をここで説明する例に適用する場合、数値的に最も高位のビットアレイを有するコードセクション204が新しいコードセクション順序の最初のコードセクションとして選択されるものと仮定する。これ以外の基準を用いて第1のコードセクションを選択することも可能である。例えば、最初のコードセクションは、無作為に選択することもできれば、プログラム開始時に実行される最初のコードセクションとなるように選択することもでき、あるいは数値的に最も下位のビットアレイとなるように選択することもできる。本発明は、このような特定の選択基準に限定されるものではない。

【0048】コードセクション204を新しいコードセクション順序の最初のコードセクションとして選択すると、コードセクション204に対応するビットアレイ“111”が現在のビットアレイになる。この現在のビットアレイを他のビットアレイと比較して、他の各ビットアレイとの間の差違度を求める。そして、現在のビットアレイとの差違度が最も小さいビットアレイに対応するコードセクションが新しいコードセクション順序における次のビットアレイとして選択される。“111”と他の全てのビットアレイとの間の差違度は、コードセクション216に対応する差違度3のビットアレイ“000”を除いて、2である。

【0049】新しいコードセクション順序におけるコードセクション202のすぐ後には1つのコードセクションしか来ることができない。そのために、同じ差違度を有するビットアレイ間の「タイ」を解消するための何らかの基準を導入しなければならない。これらのタイを解消するための一つの方法は、タイに関わるコードセクションの1つを無作為に選択することであろう。しかしながら、タイに関与するコードセクションの中で、実行期間中に最初にアクセスされたコードセクションを選択することによってより良い結果を得ることができる。ここで説明する例においては、コードセクション206が実行期間中にアクセスされた最初のコードセクションである。従って、コードセクション206が新しいコードセクション順序の2番目のコードセクションとして選択される。

【0050】コードセクション206がコードセクション順序における次のコードセクションとして選択されると、コードセクション206に対応するビットアレ「100」が現在のビットアレになる。この現在のビットアレが、まだ新しいコードセクション順序に入っていない全てのコードセクションのビットアレと比較されて、現在のビットアレとこれら他のビットアレとの間の差度が求められる。そして、現在のビットアレとの差度が最も小さいビットアレに対応するコードセクションが新しいコードセクション順序における次のビットアレとして選択される。

【0051】ここで説明する例においては、コードセクション212に対応するビットアレ「100」は現在のビットアレと同じである。従って、コードセクション212が新しいコードセクション順序の次のコードセクションとして選択される。その結果、コードセクション212に対応するビットアレ「100」が現在のビットアレになる。

【0052】この現在のビットアレが新しいコードセクション順序にまだ入っていない全てのコードセクションのビットアレと比較されて、現在のビットアレとこれら他のビットアレとの間の差度が求められる。そして、現在のビットアレとの差度が最も小さいビットアレに対応するコードセクションが新しいコードセクション順序の次のビットアレとして選択される。

【0053】ここで説明する例においては、コードセクション216に対応するビットアレ「000」が現在のビットアレとの差度が最も小さい。従って、コードセクション216が新しいコードセクション順序における次のコードセクションとして選択される。その結果、コードセクション216に対応するビットアレ「000」が現在のビットアレになる。

【0054】この現在のビットアレ「000」が新しいコードセクション順序にまだ入っていない全てのコードセクションのビットアレと比較されて現在のビットアレとこれら他のビットアレとの間の差度が求められる。そして現在のビットアレとの差度が最も小さいコードセクションが新しいコードセクション順序における次のビットアレとして選択される。

【0055】ここで説明する例においては、コードセクション202、208、210及び214に対応するビットアレは現在のビットアレとの差度が全て同じである。これらのコードセクションの中では、コードセクション208が実行期間中にアクセスされた最初のコードセクションである。そのために、コードセクション208が新しいコードセクション順序における次のコードセクションとして選択される。その結果コードセクション208に休挿するビットアレ「010」が現在のビットアレになる。

【0056】この現在のビットアレ「010」が新し

いコードセクション順序にまだ入っていない全てのコードセクションのビットアレと比較されて、現在のビットアレとこれら他のビットアレとの間の差度が求められる。そして、現在のビットアレとの差度が最も小さいビットアレに対応するコードセクションが新しいコードセクション順序における次のビットアレとして選択される。

【0057】ここで説明する例においては、コードセクション202、210及び214に対応するビットアレは現在のビットアレとの差度が全て同じである。これらのコードセクションの中では、コードセクション210が実行期間中にアクセスされた最初のコードセクションである。従って、コードセクション210が新しいコードセクション順序における次のコードセクションとして選択される。その結果、コードセクション210に対応するビットアレ「001」が現在のビットアレになる。

【0058】この現在のビットアレ「001」が新しいコードセクション順序にまだ入っていない全てのコードセクションのビットアレと比較されて、現在のビットアレとこれら他のビットアレとの間の差度が求められる。そして、現在のビットアレとの差度が最も小さいビットアレに対応するコードセクションが新しいコードセクション順序における次のビットアレとして選択される。

【0059】ここで説明する例においては、コードセクション202及び214に対応するビットアレは現在のビットアレと同じである。これらのコードセクションの中では、コードセクション202が実行期間中にアクセスされた最初のビットアレである。従って、コードセクション202が新しいコードセクション順序における次のコードセクションとして選択される。その結果、コードセクション202に対応するビットアレ「001」が現在のビットアレになる。

【0060】この現在のビットアレ「001」が新しいコードセクションにまだ入っていない全てのコードセクションのビットアレと比較されて、現在のビットアレとこれら他のビットアレとの間の差度が求められる。そして、現在のビットアレとの差度が最も小さいビットアレに対応するコードセクションが新しいコードセクション順序における次のビットアレとして選択される。

【0061】この時点になると、コードセクション214が新しいコードセクション順序に入っていない唯一のコードセクションになる。その結果、コードセクション214が新しいコードセクション順序における次のコードセクションとして選択される。これで、全てのコードセクションが新しいコードセクション順序に並べられて、順序付けプロセスは終了する。

【0062】図6は、コードセクションが上記の「最小

差速度」順序付けプロセスの結果により順序付けられたコンピュータプログラムPの命令データイメージ600を示す。命令データイメージ500の場合と同様に、図示の命令データイメージ600のコードセクションの順序は、元の命令データイメージ200におけるコードセクションの順序に比べて、コンピュータプログラムPによって実行される高レベルオペレーションに対してより強い相関を有する。

【0063】上には、実行期間中にコードセクションが何時アクセスされるかに基づいてコードセクションを順序付ける2つの方法を説明した。しかしながら、上に説明した方法は、単に例示的なものである。本発明においては、これら以外にもいろいろな方法を用いることができる。例えば、コードセクションをそれらのコードセクションが実行期間中に最初にアクセスされる順序を反映するよう順序付けることもできる。ここで説明する例にこの方法を適用すると、コードセクションは次の順序により順序付けされる。コードセクション206、コードセクション204、コードセクション212、コードセクション208、コードセクション210、コードセクション202、コードセクション214、そして最後にコードセクション216。この順序はアクセスレコードから容易に決定することができ、アクセスマトリクスの構築を必要としないということに留意すべきである。

【0064】命令データイメージの順序付け  
コンピュータプログラムのコードセクションの新しい順序が決定されたならば、その新しいコードセクション順序を反映するようコードセクションが再配列された新しい命令データイメージの構築が行われる。本発明の一実施形態によれば、新しい命令データイメージはマップファイルに基づいて生成される。具体的に言うと、F1、F2、F3、F4、F5、F6、F7及びF8の各関数に対応するオブジェクトコードファイルのリンク順序を指定するマップファイルが生成される。各関数に対して指定されるリンク順序は、それらの関数に対応するコードセクションについて決定された順序に基づいて指定される。従って、図6に示す命令データイメージを構築するためには、マップファイルは、オブジェクトコードファイルがF2、F3、F6、F8、F4、F5、F1、F7の順序に従ってリンクすべきであるということ指定する。このマップファイルはプロセッサ102で実行中のリンカープログラムによって読み取られる。リンカープログラムは、マップファイルに基づいて、プロセッサ102に関数F1、F2、F3、F4、F5、F6、F7及びF8に対応する種々のオブジェクトコードモジュールをリンクさせて、図6に示す命令データイメージ600を構築する。

【0065】本発明により決定されるコードセクション順序に基づいてコードセクションが順序付けられた命令データイメージを構築するには、他の方法を用いること

もできる。例えば、コンパイラマップファイルを読み取るためのコンパイラを構築することもできる。その場合、コンパイラマップファイルは、関数または関数の部分に対してオブジェクトコードが生成される順序を指定する。あるいは、現命令データイメージを単に再配列するだけで新しい命令データイメージを形成するメカニズムを構築することもできる。例えば、プロセッサ102は、命令データイメージ200からコードセクションを抽出して、命令データイメージ600を生成するための新しい順序に従い記憶するようプログラムすることができ、この再配列プロセスは、異なるコードセクション内における一部のアドレス参照を参照されるコードの新しい位置を反映するよう修正することが必要な場合もある。

【0066】以上の説明においては、コードセクションとの関連で順序付け方法を特定の説明した。しかしながら、本発明はコードセクションの順序付けに限定されるものではない。具体的に言うと、上に説明した技術的方法は、プログラムのコードを順序付ける代わりに、コンピュータプログラムの実行中にアクセスされるデータを順序付けるために用いることもできる。例えば、データベースプログラムにおいては、プログラムはデータの一部のセクションに他のセクションとは異なるタイミング及び周波数でアクセスすることが可能である。これらのデータアクセスは、データのセクションがアクセスされるのと同じ順序で記憶されている場合により効率的である。この順序付け方法をデータに適用するには、データベースプログラムは実行期間中に実行される。そして、実行期間中にデータベースプログラムによって行われたデータアクセスが検出され、記録される。このように記録されたアクセス情報を用いて、データが上記の順序付け方法に従い順序付けられる。

【0067】プログラムされた「オブジェクト」は、関連付けられた関数及びデータの集合である。上記の技術的方法を用いて全コードまたは全データのセクションを順序付けることができるのとまったく同様に、上記の方法は、コードとデータをどちらも含むオブジェクトを順序付けるために用いることもできる。

【0068】

【発明の効果】本願で説明した順序付け方法を用いることによって、明確かつ顕著な2つの効果を達成することができる。第1に、同じ動作に対応するルーチンのいくつかのコード部分をより緊密な関係にまとめることができる。所与の動作に対するコードがより少ないコードセクションにまたがって分布することになるため、所与の動作の実行時に高速メモリに常駐せねばならない平均コード量が少なくなる。その結果、同じ動作をより少ない高速メモリ容量を用いて実行することができる。これによって、より少ない高速メモリを持つシステムは、順序付けられたコードを、より多量の高速メモリが

順序付けられていないコードを実行するのと同じ効率レベルで実行することができる。さらに、他の同時実行される動作のためにより多くの高速メモリが使用可能になる。

【0069】もう一つの効果は、動作を逆行するのに必要なコードセクションをローディングするとき起こるページフォルトの平均数が著しく少なくなる。ページフォルト数が少なくなるのは、コードセクションが使用される順にそろえられるためである。従って、いっしょに用いられ、いっしょにロードされるコードセクションは一つまとめて記憶される。

【0070】以上、本発明の特定の実施形態について詳細に説明したが、本願によって様々な修正態様及び変形態様は自明であろう。そのような修正態様並びに変形態様は、特許請求の範囲に記載するところに従い本発明の範囲内に包含されるものである。

【図面の簡単な説明】

【図1】 本発明を実装することができるコンピュータシステムのブロック図である。

【図2】 一例のコンピュータプログラムの異なるコードセクションを示す説明図である。

【図3】 図2に示すコンピュータプログラムの異なるコードセクションが実行期間中に何時アクセスされるかを示すタイミング図である。

【図4】 図2に示す実行期間中に記録されたアクセスに基づいて構成されたアクセスマトリクスを示す説明図である。

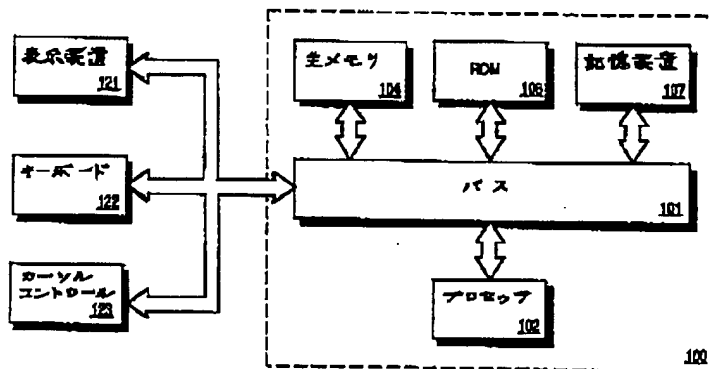
【図5】 図2に示すコンピュータプログラムのコードセクションが、本発明の一実施形態に従って記録された新規な命令データイメージを示す説明図である。

【図6】 図2に示すコンピュータプログラムのコードセクションが、本発明の一実施形態に従って記録された新規な命令データイメージを示す説明図である。

【符号の説明】

100 コンピュータシステム、101 バス、102 プロセッサ、104主メモリ、106 ROM、107 記憶装置、121 表示装置、122 キーボード、123 カーソル制御。

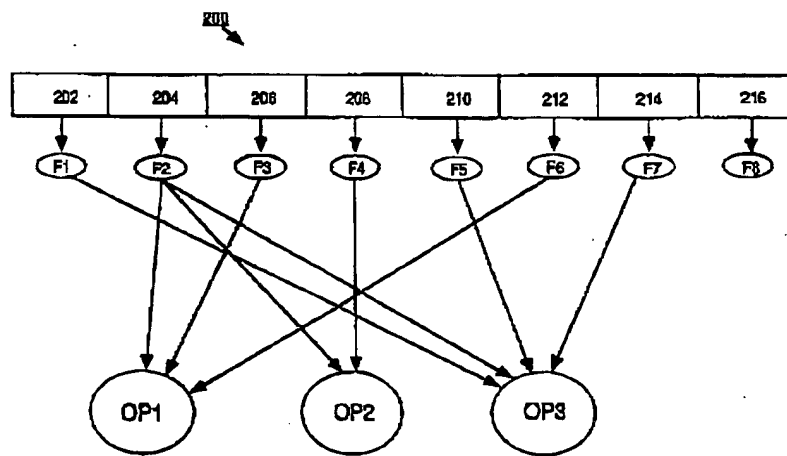
【図1】



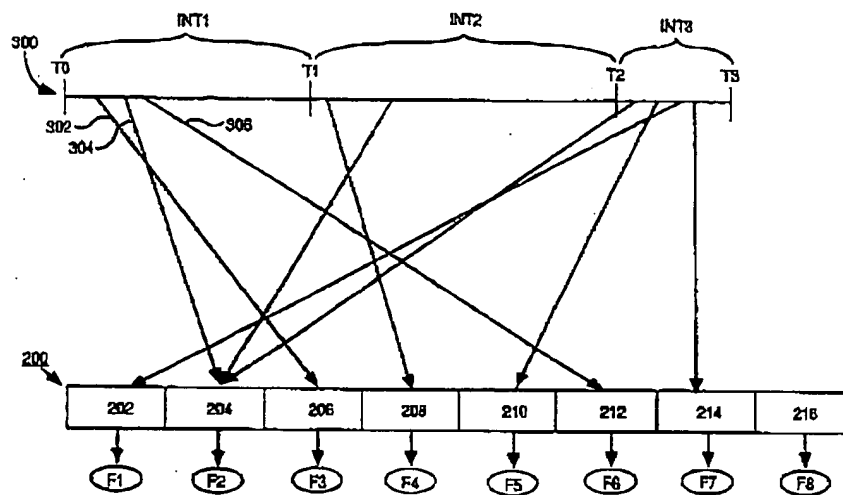
【図4】

	402	404	406	400
202	0	0	1	408
204	1	1	1	410
206	1	0	0	412
208	0	1	0	414
210	0	0	1	416
212	1	0	0	418
214	0	0	1	420
216	0	0	0	422

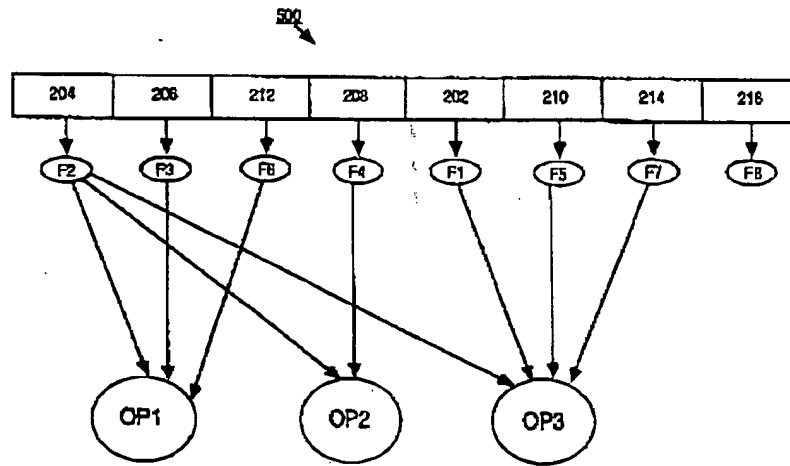
【図2】



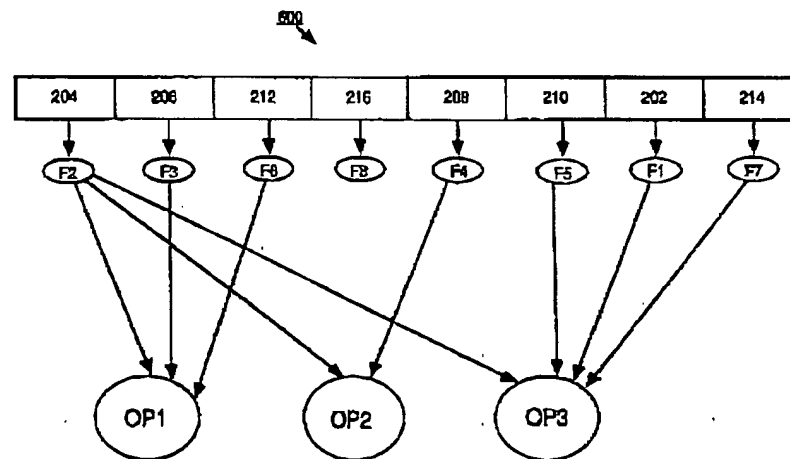
【図3】



【図5】



【図6】



フロントページの続き

(72)発明者 ケビン・ジェイ・クラーク  
 アメリカ合衆国 95125 カリフォルニア  
 州・サン ホゼ・カーティス アヴェニ  
 ュ・1376